Asymptotics of Replication and Matching in Large Caching Systems

Arpan Mukhopadhyay, Nidhi Hegde, and Marc Lelarge

***Abstract*— We consider a generic model of distributed caching systems, where the cache servers are constrained by two main resources: memory size and bandwidth. Content distribution networks (CDNs) providing video contents and peer-to-peer video-on-demand services are a few examples of such systems. The throughput of these systems crucially depends on how these resources are managed, i.e., how contents are replicated across servers and how requests of specific contents are matched to servers storing the contents. In this paper, we formulate the problem of computing the replication policy and the matching policy, which jointly maximizes the throughput of the caching system. It is shown that computing the optimal replication policy for a given finite system is an NP-hard problem. A greedy replication scheme is then proposed and is shown to achieve a constant factor approximation guarantee when combined with the optimal matching policy. We note that the optimal matching policy has the problem of interruption in service of the ongoing requests due to re-assignment or repacking of the existing requests. To avoid this problem, we propose a simple randomized online matching scheme and analyze its performance in conjunction with the proposed replication scheme. We consider a limiting regime, where the number of servers is large and the arrival rates of the contents are scaled proportionally, and show that the proposed policies achieve asymptotic optimality. Extensive simulation results are presented to evaluate the performance of different policies and study the behavior of the caching system under different service time distributions of the requests.**

***Index Terms*— Caching, content delivery network, replication, matching, mean field limit, differential inclusion (DI).**

I. Introduction

RECENT explosive growth in Internet traffic, stemming mainly from the transfer of multi-media contents, e.g., streaming videos, movies, has led to the emergence of content distribution networks (CDNs) and peer-to-peer systems that support the demand for popular contents by replicating the contents at the network periphery (e.g., boxes or servers). It is expected that video streaming services and downloads will account for more than 81% of all Internet's traffic by 2021 [2]. To support this increasing demand, video-streaming services such as Netflix, Youtube often use CDN's to serve the requests of their most popular contents.

Large CDN's usually consist of a central server, storing an entire catalogue of contents, and a large number of edge servers, storing a small fraction of popular contents in their caches and serving requests of the stored contents [3]. In such systems, it is assumed that access to the central server is expensive. Therefore, a large portion of the content requests must be served by the edge servers that are constrained by their limited memory and bandwidth capacities. In this paper, we model these servers as loss servers [4] and aim at minimizing the number of requests blocked at these servers (and thus need to be sent to the central server). We do not consider queuing of the requests since we focus on delay-sensitive services, which comprise a large proportion of the Internet's traffic today [2].

Efficiency of such systems crucially depends on the replication (also called allocation) policy used to populate the caches of the servers and the request matching policy used to dispatch the incoming requests. In this paper, we analyze the effect of both replication and matching policies on distributed caching systems consisting of a large number of cache servers and a central request router [5], [6]. Specifically, we make the following contributions:

(1) *Formulation of the joint problem*: We first formulate the problem of computing the optimal allocation policy, which, if combined with the optimal matching policy (maximum matching), maximizes the number requests served per unit time by the caching system in the stationary regime. To the best of our knowledge, this is the first work that addresses this joint allocation-matching problem. We show that the joint problem for finite systems is an NP-hard. A polynomial-time greedy allocation scheme is proposed and is shown to achieve a constant factor approximation guarantee.

(2) *Online matching scheme*: Next, we turn our attention to the dynamics of the system, which is determined by the matching policy in use. In earlier works, e.g., [7], [8] a maximum matching scheme has been considered to match incoming requests to servers. The maximum matching scheme requires the knowledge of the states of all the servers in the system and causes reassignment of existing requests when a new request joins the system. Such interruption in service is clearly not desired and may cause significant delay in serving the requests. We thus propose a simple randomized online policy for request matching which does not cause interruption to the already existing requests and which can be implemented with a small number of asynchronous messages from the servers to the job dispatcher.

Manuscript received August 5, 2018; revised February 27, 2019 and June 19, 2019; accepted June 24, 2019; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Paschos. An earlier version of this work [1] has been presented in the 2018 IEEE INFOCOM Conference. *(Corresponding author: Arpan Mukhopadhyay.)*

A. Mukhopadhyay is with the Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K. (e-mail: arpan.mukhopadhyay@warwick.ac.uk).

N. Hegde is with Borealis AI, Edmonton, AB T5J 3G2, Canada (e-mail: nidhi.hegde@borealisai.com).

M. Lelarge is with INRIA, 75589 Paris, France, and also with ENS, 75589 Paris, France (e-mail: marc.lelarge@ens.fr).

Digital Object Identifier 10.1109/TNET.2019.2926235

(3) *Analysis of the dynamics*: We analyze the dynamics of the caching system operating under the proposed greedy replication scheme and a previously studied [7], [9] 'proportional-to-product' replication scheme in conjunction with the proposed online matching scheme. We show that these combinations are asymptotically optimal in the limiting regime where the number of cache servers and request arrival rates scale proportionally with each other and the number of contents remains fixed. Such a scaling regime corresponds to scenarios where a fixed number of most popular contents are served by a large number of cache servers. The proof of asymptotic optimality uses fluid limits of processes describing the dynamics of the system. The fluid limit in our case cannot be described by ordinary differential equations (ODE's) since it describes a non-smooth dynamical system. The novelty in our approach lies in describing the fluid limit as a solution to a differential inclusion (DI) system and showing that all trajectories of the solutions converge to the same global attractor.

(4) *Simulation results*: We provide extensive simulation results to analyze the performance of the proposed schemes. In particular, we show that for large systems the system performance is close to the optimum as predicted by our theoretical results. We further show that under the proposed online matching policy and for large system sizes the system performance is nearly insensitive to service time distributions so long as the mean of the distribution remains the same.

Related Work: Content placement in caching systems has been the subject of study for many years now. Of the numerous papers on this topic, we mention a few that are most relevant to our work. In [9]–[11], optimal cache allocation policy was designed without taking into account the bandwidth restrictions of the servers. Joint request routing and caching problems in the context wireless networks have been considered in [11]–[15]. However, in all these studies the objective is to either minimize the delay in serving requests or to minimize the cost of using network resources. A loss model of caching systems was first introduced in [7] in the context of peer-to-peer video on demand services. A 'proportional-to-product' replication policy in which the number of replicas is proportional to the arrival rate of the contents was proposed and analyzed in conjunction with the maximum matching algorithm. In [8] a similar loss model was considered. However, the objective was to maximize the utilization of the resources as opposed to maximization of throughput. In [16], a discrete-time model similar to ours is considered. The objective there is to minimize the expected transmission rate from the main server in order to serve all requests in a time slot. A different scaling regime in which the number of contents is scaled is considered.

Once the allocation of contents to the caches has been fixed, the problem of assigning incoming requests to the servers optimally is equivalent to the well-known maximum matching problem on bipartite graphs. However, unlike the classical maximum matching problem, in our setting the matching decisions are irrevocable. Therefore, we focus on online matching algorithms. Online bipartite matching algorithms such as the ranking algorithm in [17] or the more general greedy algorithm in [18] have been analyzed under an idealized setting where n balls arrive sequentially into n bins and at each arrival the edges between the arriving ball and the bins are chosen independently at random. These algorithms guarantee an average matching of size $1-1/e$ of the optimal matching. However, in our setting the edges between each arriving content request and the servers are governed by the content replication scheme and are not independent across different arrivals. This completely changes the analysis and the optimality results. Another algorithm similar to the RAS was analyzed in [19] for a cloud-based systems. However, there the requests can be placed to any server as long as the server has enough bandwidth. This is different from our setting where we additionally have memory constraints at each server.

Organization: The remainder of the paper is organized as follows. Section II introduces the system model. In Section III we formulate the joint allocation and matching problem and analyze the complexity of the problem. In Section IV we present efficient approximate algorithms. In Section V we present a randomized online matching policy and analyze the system performance under this policy in the large-systems asymptotic regime. Section VI presents simulation results. Finally, the paper is concluded in Section VII.

II. System Model

We consider a dynamic model of caching systems in which requests for contents arrive at random instants and are served by servers storing the corresponding contents. The servers are assumed to be able to serve only a finite number of requests simultaneously.

A. Server and Storage Model

The caching system consists of n servers and m contents indexed by the sets $S=\{1,2,\ldots,n\}$ and $C=\{1,2,\ldots,m\}$, respectively. We assume that each server $s\in S$ is capable of storing up to $d_s\geq 1$ contents in its cache and has a bandwidth of $U_s>0$, i.e., it can serve at most U_s requests simultaneously. The *replication*, or *allocation* policy is represented by a binary matrix $A=(a_{sc})_{s\in S,c\in C}\in\{0,1\}^{nm}$, with $a_{sc}=1$ if c is stored in the cache of s and $a_{sc}=0$, otherwise. Thus, for any *feasible replication policy* $A=(a_{sc})_{s\in S,c\in C}$, we have

$$\sum_{c\in C} a_{sc}\leq d_s \quad \text{for all } s\in S \tag{1}$$

The set of all feasible replication policies is denoted as $\mathcal{A}\subset\{0,1\}^{nm}$. The cache of each server is populated at $t=0$ according to some cache allocation policy $A\in\mathcal{A}$ and is kept unchanged for all $t\geq 0$. Servers are assumed to have different bandwidths and memory sizes from the sets $\{U_i, i\in\mathcal{I}\}$ and $\{d_j, j\in\mathcal{J}\}$, respectively, where $\mathcal{I}$ and $\mathcal{J}$ are some finite index sets. The fraction of servers having bandwidth U_i and memory size d_j is denoted as α_{ij} for each $(i,j)\in\mathcal{I}\times\mathcal{J}$.

B. Service Model

Requests of contents are assumed to arrive one at a time according to simple point processes. The average number of

requests of a content $c \in C$ arriving per unit time is denoted by λ_c and is called the *arrival rate* or the *popularity* of the content. The vector of content popularities is denoted as $\Lambda = (\lambda_c, c \in C)$ and is assumed to be a known constant throughout the paper. Typically, the popularities of items containing videos or movies vary over periods ranging from few hours to few weeks [10] (and have to be estimated periodically [20]) which are slower than the time scales of interest in the paper. For this reason we do not consider the effect of time-varying popularities and errors in estimation of the popularities in this paper.

Each arriving request is *matched* or *assigned* to a server in the system according to some matching scheme. For a request to be matched to a server, the server must be *available*, i.e., it must store the content in its cache and must have available bandwidth to process the request. If upon arrival of a request no such server is available, then the request cannot be served and is said to be *blocked* or dropped by the caching system. In this case the request is immediately routed to a central server that stores all the contents. The accepted requests are assumed to stay in the system for a random amount of time, independent and identically distributed (i.i.d) with unit mean. Our goal is to maximize the fraction of accepted requests or to minimize the fraction of blocked requests in the system. For analytical convenience, we assume that each request can be served by at most one server simultaneously. We note that an analysis similar to the one presented in this paper can be carried out for the case where multiple servers are allowed to serve the same request. However such an analysis is requires additional assumptions on the model and is beyond the scope of the current paper.

III. Problem Formulation

Our first objective is the design of an allocation policy which populates the caches at the start of the system. We seek an allocation A that maximizes the average number requests served per unit of time by the caching system operating in the stationary regime. To formulate the problem, we consider a time window of unit length. Let X_c, $c \in C$ denote the number of requests of content c arriving in this window and define $X := (X_c, c \in C)$.

Here we make the following approximations: (1) The cache servers are idle at the beginning of each unit time window. (2) All requests that are accepted by the system in a unit time window stay in the system exactly till the end of the time window. These approximations are required to formulate a problem without requiring assumptions on the arrival and departure processes. Both these approximations are exact for a slotted time system where arrivals occur at the beginning of each time slot and leave at the end of the time slot. The approximation of the continuous-time system with a slotted-time system is accurate for large systems due to the mean field behavior of such systems. The detailed dynamics in continuous-time under specific assumptions on arrival processes and service time distributions are analyzed in Section V.

Let $b_{sc} \in \mathbb{Z}_+$ denote the total number of requests of content $c \in C$ matched/assigned to a server $s \in S$ in a given unit time window. Then, the *matching* $B = (b_{sc})_{s\in S, c\in C}$ must satisfy

$$\sum_{c\in C} b_{sc} \leq U_s, \quad \forall s \in S, \tag{2}$$

$$\sum_{s\in S} b_{sc} \leq X_c, \quad \forall c \in C, \tag{3}$$

$$\mathbb{1}\left(b_{sc} > 0\right) \leq a_{sc}, \tag{4}$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. Let $\mathcal{B}(A, X)$ and $\mathcal{B}(X)$ denote the set of matchings B satisfying constraints (2), (3), (4) and constraints (2), (3), respectively. Thus, under a given allocation policy A, the maximum total number of requests that can be matched in this time window is given by

$$M(A, X) = \max_{B\in\mathcal{B}(A,X)} \sum_{s\in S c\in C} b_{sc} = \max_{B\in\mathcal{B}(X)} \sum_{s\in S c\in C} a_{sc} b_{sc}.$$

Given the allocation A and the demand vector X, finding $M(A, X)$ is equivalent to finding a maximum matching in a bipartite graph and thus can be done in polynomial time. However, our goal is to find an allocation policy A for which $M(A, X)$ is maximized, i.e., we aim to find a solution to

$$\max_{A\in\mathcal{A}} M(A, X) = \max_{A\in\mathcal{A}} \max_{B\in\mathcal{B}(X)} \sum_{s\in S, c\in C} a_{sc} b_{sc}. \tag{5}$$

We note that in the above formulation the vector X is random and not known *a priori*. Since our goal is to find a solution which works well on average, we replace the objective function by its expected value taken over the distribution of the random vector X. Hence, our interest is to look for solutions of the following problem:

$$\max_{A\in\mathcal{A}} \mathbb{E}\left[\max_{B\in\mathcal{B}(X)} \sum_{s\in S, c\in C} a_{sc} b_{sc}\right], \tag{6}$$

where the expectation is taken with respect to the distribution of the random demand vector X. It is difficult to obtain an analytically tractable expression of the expectation in (6) due to the complex structure of the space $\mathcal{B}(X)$. To overcome this difficulty, we use a method called the sample average approximation (SAA) [21] to replace the expectation with the average over T independent samples of the demand vector X. Hence, we consider the following problem:

$$\begin{aligned}\max_{A\in\mathcal{A}} \frac{1}{T} \sum_{t=1}^{T} \max_{B^{(t)}\in\mathcal{B}(X^{(t)})} \sum_{s\in S} \sum_{c\in C} a_{sc} b_{sc}^{(t)} \\ = \max_{A\in\mathcal{A}} \max_{B^{(t)}\in\mathcal{B}(X^{(t)}),\forall t} \sum_{s\in S} \sum_{c\in C} a_{sc} \left(\frac{1}{T} \sum_{t=1}^{T} b_{sc}^{(t)}\right),\end{aligned} \tag{7}$$

where $X^{(t)} = (X_c^{(t)}, c \in C)$ denotes the t^{th} sample of the demand vector and $B^{(t)} = (b_{sc}^{(t)})_{s\in S, c\in C} \in \mathcal{B}(X^{(t)})$ for all $t = 1 : T$. The interchange of max and $\sum$ holds since $a_{sc}, b_{sc}^{(t)} \geq 0$ for all $t = 1 : T$. It is shown in Proposition 2.1 of [21] that as $T \to \infty$, the optimal solution of (7) approaches the optimal solution of (6). Hence, this approximation is accurate for large enough T.

Finally, we consider a relaxed version of problem (7) where instead of satisfying the constraint $B^{(t)} \in \mathcal{B}(X^{(t)})$ for every t,

we require only the average matching $\bar{B} = (\bar{b}_{sc})_{s\in S, c\in C} = \frac{1}{T}\sum_{t=1}^{T} B^{(t)} \in \mathbb{R}_{+}^{n\times m}$ to satisfy $\bar{B} \in \mathcal{B}(\bar{X}(T))$, where $\bar{X}(T) = \left(\bar{X}_c(T) = \frac{1}{T}\sum_{t=1}^{T} X_c^{(t)}, c \in C\right)$, denotes the average demand vector. We note that the entries $\bar{b}_{sc}$ of the average matching $\bar{B}$ are non-negative reals as opposed to the entries of B that are non-negative integers. This relaxation is required since $X^{(t)}$ is random and not known a priori. Thus, we have the following optimization problem:

$$\max_{A\in\mathcal{A}} M(A, \bar{X}(T)) = \max_{A\in\mathcal{A}} \max_{\bar{B}\in\mathcal{B}(\bar{X}(T))} \sum_{s\in S, c\in C} a_{sc}\bar{b}_{sc} \quad (8)$$

We note that $\bar{X}(T) \to \Lambda = (\lambda_c, c \in C)$ as $T \to \infty$. Hence, for large T the problem is given by

$$\begin{aligned}\max_{A\in\mathcal{A}} M(A, \Lambda) &= \max_{A\in\mathcal{A}} \max_{\bar{B}\in\mathcal{B}(\Lambda)} \sum_{s\in S, c\in C} a_{sc}\bar{b}_{sc} \\ &= \max_{A\in\mathcal{A}} \max_{\bar{B}\in\mathcal{B}(A,\Lambda)} \sum_{s\in S, c\in C} \bar{b}_{sc} \quad (9)\end{aligned}$$

We refer to the problem above as the joint allocation-matching (JAM) problem since in it the decision variable A also affects the average matching $\bar{B} \in \mathcal{B}(A, \Lambda)$. Our first result establishes the following equivalence.

Proposition 1: Problem (9) *is equivalent to the following problem*

$$\begin{aligned}&\underset{Z=(z_{sc})_{s\in S, c\in C}}{\textit{Maximize}} \sum_{s\in S}\sum_{c\in C} z_{sc} \\ &\textit{subject to} \sum_{c\in C} z_{sc} \le U_s, \quad \forall s \in S \\ &\sum_{s\in S} z_{sc} \le \lambda_c, \quad \forall c \in C \\ &\sum_{c\in C} \mathbb{1}(z_{sc} > 0) \le d_s, \quad \forall s \in S \\ &z_{sc} \in \mathbb{R}_{+}, \quad \forall s \in S, \ \forall c \in C \quad (10)\end{aligned}$$

Furthermore, if $Z^* = (z_{sc}^*)_{s\in S, c\in C}$ is an optimal solution of (10)*, then an optimal solution A^* of problem* (9) *can be found by setting $a_{sc}^* = \mathbb{1}(z_{sc}^* > 0)$, for all $(s, c) \in S \times C$.*

Proof: Let $Z = (z_{sc})_{s\in S, c\in C}$ be a feasible solution of (10). Set $a_{sc} = \mathbb{1}(z_{sc} > 0)$ and $\bar{b}_{sc} = z_{sc}$ for all $s \in S$, $c \in C$. Then clearly we have $\bar{B} = (\bar{b}_{sc})_{s\in S, c\in C} \in \mathcal{B}(A, \lambda)$. Furthermore, $\sum_{s\in S}\sum_{c\in C} z_{sc} = \sum_{s\in S}\sum_{c\in C} \bar{b}_{sc} \le O_1$, where O_1 denotes the optimal value of problem (9). Taking the maximum of the LHS over the set of feasible solutions of (10) (this is possible since the feasible set of solutions is compact and the objective function is continuous) we have $O_2 \le O_1$, where O_2 denotes the optimal value of problem (10).

Conversely, suppose that $A = (a_{sc})_{s\in S, c\in C} \in \mathcal{A}$, $\bar{B} = (\bar{b}_{sc})_{s\in S, c\in C} \in \mathcal{B}(A, \lambda)$. For all pairs $(s, c) \in S \times C$ such that $a_{sc} = 0$ we set $z_{sc} = 0$. For all other pairs $(s, c) \in S \times C$ we set $z_{sc} = \bar{b}_{sc}$. Clearly, $(z_{sc})_{s\in S, c\in C}$ is a feasible solution of problem (10). Moreover, we have $\sum_{s\in S}\sum_{c\in C} \bar{b}_{sc} = \sum_{s\in S}\sum_{c\in C} a_{sc}\bar{b}_{sc} = \sum_{s\in S}\sum_{c\in C} z_{sc} \le O_2$, Now taking the maximum of the LHS over all feasible solutions of problem (9) we obtain $O_1 \le O_2$. Hence, we have $O_1 = O_2$. The first part of the proof also shows how to construct an optimal solution of (9) from that of (10). □

Theorem 1: Problem (10) *is NP-hard.*

The proof of Theorem 1 uses the equivalent form (10) and is given in Appendix A.

IV. Replication Algorithms

Since the JAM is NP-hard, an efficient algorithm for finding an exact solution is out of reach (unless $P = NP$). We therefore look for replication/allocation algorithms which provide approximate solutions and are easy to implement. Specifically, we consider the following allocation policies:

1) The `Greedy` Policy: The `greedy` algorithm iteratively computes a feasible replication policy $A \in \mathcal{A}$ by assigning, in each iteration, a flow to each server $s \in S$ and each content $c \in C$, denoted as $\mathsf{flow}(s)$ and $\mathsf{flow}(c)$, respectively. Additionally, it assigns to each server $s \in S$ a degree, denoted as $\mathsf{deg}(s)$, in each iteration. Initially, we set $\mathsf{flow}(s) = U_s$, $\mathsf{deg}(s) = d_s$ for all $s \in S$ and $\mathsf{flow}(c) = \lambda_c$ for all $c \in C$. Then, in each iteration, a pair $(s, c) \in S \times C$ that maximizes $\min(\mathsf{flow}(s), \mathsf{flow}(c))$ and for which $\mathsf{flow}(s)\mathsf{deg}(s)\mathsf{flow}(c) > 0$ is found. If (s^*, c^*) denotes such a pair then the flow of both s^* and c^* are decreased by an amount $\texttt{MatchedFlow} = \min(\mathsf{flow}(s^*), \mathsf{flow}(c^*))$ and the degree of s^* is reduced by one. Furthermore, we set $a_{s^*c^*} = 1$ and $z_{s^*c^*} = \texttt{MatchedFlow}$. The iterations continue until no such pair (s^*, c^*) can be found. The pseudocode of the algorithm is given as Algorithm 1.

Algorithm 1 `greedy`$(\mathcal{U}, \Lambda, \mathcal{D})$

Inputs: $\mathcal{U} = (U_s, s \in S)$, $\Lambda = (\lambda_c, c \in C)$, $\mathcal{D} = (d_s, s \in S)$
Output: $Z = (z_{sc})$, $A = (a_{sc})$
Initialize: $\mathsf{flow}(s) \leftarrow U_s, \mathsf{deg}(s) \leftarrow d_s, \forall s \in S$; $\mathsf{flow}(c) \leftarrow \lambda_c, \forall c \in C$

```
while ∃(s, c) ∈ S × C s.t flow(s)deg(s)flow(c) > 0 do
    s* ← arg max_{s∈S:deg(s)>0} (flow(s))
    c* ← arg max_{c∈C} (flow(c))
    MatchedFlow ← min(flow(s*), flow(c*))
    flow(s*) ← flow(s*) − MatchedFlow, flow(c*) ← flow(c*) − MatchedFlow, deg(s*) ← deg(s*) − 1
    a_{s*c*} ← 1, z_{s*c*} ← MatchedFlow
end while
```

Thus, in the `greedy` replication policy, three factors are taken into consideration to store a content in the cache of a server. These are: (1) the popularity of the content, (2) the bandwidth capacity of the server, and (3) the memory size of the server. The `greedy` algorithm proceeds by selecting in each iteration the 'best' (s, c) pair in terms of the above three criterions. In particular, the `greedy` algorithm selects the server with the maximum remaining bandwidth capacity among all the servers having non-zero remaining memory size to store a content having the current highest popularity.

Clearly, the `greedy` algorithm terminates in at most $m + n$ iterations and returns a feasible replication policy in $\mathcal{A}$. Furthermore, in each iteration, the optimal pair (s^*, c^*) can be found in at most $O(m + n)$ steps. Thus, the worst case time complexity of the `greedy` algorithm is $O((m+n)^2)$. We refer

to the allocation policy computed by the greedy algorithm as the greedy allocation policy.

We now show that the greedy algorithm always achieves at least $1/2$ of the optimal value of problem (10). To state the result, we denote the instance of problem (10), defined by the vector of bandwidth capacities $\mathcal{U} = (U_s, s \in S)$, vector of arrival rates $\Lambda = (\lambda_c, c \in C)$, and the vector of memory sizes $\mathcal{D} = (d_s, s \in S)$ by $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$ and its optimal value by $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$. The following theorem, whose proof is given in Appendix B, provides a performance guarantee for the greedy algorithm.

Theorem 2: The output of greedy *on* $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$ *is at least* $\frac{1}{2}\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$.

Remark 1: We note that without the constraint given by the indicator function, problem (10) is equivalent to the fractional b-matching problem [22], [23] which is solvable exactly in polynomial time for bipartite graphs. However, the non-linear constraint given by the indicator function makes (10) an NP-hard problem. Both the fractional b-matching problem and problem (10) fall under the broad class of combinatorial problems that can be described by *subset systems* or *independence systems* [24], [25]. For independence systems there is a well-developed general theory (e.g., Theorem 1 of [25]) which shows that a greedy algorithm for such problems always returns a q-approximate solution where q is the *rank quotient* of the independence system under consideration. The value of q, however, depends on the specific system under consideration and has to be derived for each system individually. In the proof of Theorem 2, we essentially show that $q \geq 1/2$ for problem (10). We make the proof self-contained so that no prior knowledge of the theory of independence systems is required to understand the proof.

*2) Proportional to Product (*p2p*) Policy:* We consider another randomized policy for allocating contents to servers. Under this policy, a server with memory size d_j, $j \in \mathcal{J}$, is allocated all contents belonging to a set $K \subseteq C$ of size d_j with probability

$$p_{jK} = \frac{1}{Z_j} \prod_{c \in K} \hat{\lambda}_c, \tag{11}$$

independently of all other servers in the system, where $\hat{\lambda}_c = \lambda_c / \sum_{c' \in C} \lambda_{c'}$ denotes the normalized arrival rate of content c and $Z_j = \sum_{K \subseteq C, |K| = d_j} \prod_{c \in K} \hat{\lambda}_c$,. This scheme was proposed in [7]. Unlike the greedy policy, it does not take into account the bandwidths of the servers. Furthermore, unlike the greedy scheme, it is difficult to provide any performance guarantee for this scheme for finite system sizes. Nevertheless, we analyze the dynamics and evaluate the performance of the system under the p2p policy later in the paper.

Remark 2: Although we assume the content popularities to be known (or estimated) in advance, we note that the p2p scheme need not require such prior estimates since it can automatically learn content popularities. This can be achieved as follows: whenever request of a new content $c \in C$ arrives, with probability $\sum_{i \in \mathcal{I}} \alpha_{ij}$ a server of memory size d_j is chosen uniformly at random from all the servers with memory size d_j. If content $c \in C$ already exists in the cache of the chosen server then do nothing, otherwise, remove a content, chosen uniformly at random from the cache, and replace it by content c. It can be easily shown that under this cache replacement scheme the stationary distribution of the cache configuration is given by (11).

*3) Uniform (*unif*) Policy:* As a baseline for comparison, we consider a naive strategy for replication where a server with memory size d_j, $j \in \mathcal{J}$, is populated by all the contents of the set $K \subseteq C$ of size d_j with probability

$$p_{jK} = \frac{1}{\binom{m}{d_j}} \tag{12}$$

Note that (12) does not depend on the particular set K and is the same for all K of the same size. Furthermore, (12) follows from (11) if $\hat{\lambda}_c = 1/m$ for all $c \in C$. Thus, the unif policy treats all contents to be equally popular. It is easy to implement this policy in practice since it does not require the knowledge of the popularities of the contents.

V. Matching Algorithms: Dynamics of the System

We now analyze the dynamics of the system when the caches have been populated by one of the algorithms discussed in the previous section. The dynamics of the system is primarily governed by the matching scheme being used by a central router/dispatcher making the routing decisions of the incoming jobs [5], [6]. The optimal matching scheme has been considered in previous works e.g. [7], [8]. In this scheme, an incoming request is accepted if a matching $B = (b_{sc})_{s \in S, c \in C}$ satisfying the constraints (2) and (4) can be found such that $\sum_{s \in S, c \in C} b_{sc}$ equals the total number of requests in the system including the new request. Such a matching, although optimal, (1) requires the knowledge of the state of each server in the system and (2) involves *repacking* or reassignment of the ongoing requests when a new request joins the system. For large caching systems, obtaining state information of all servers is expensive and is often not feasible. Furthermore, repacking of requests causes undesirable interruptions in service and leads to significant delays. We therefore look for a matching policy that is simpler to implement and does not involve repacking of the ongoing requests.

A. Random Available Server (RAS) Matching Policy

In this policy, each newly arrived content request is assigned to a server chosen uniformly at random from the set of all servers that store the content and are able to serve an additional request of the content. If no such server is available, then the request is blocked. This scheme can be implemented by maintaining a list of available servers for each content at the central job dispatcher. It is not necessary for the central job dispatcher to keep track of the number of jobs in each server. It is sufficient to just to know if a server can process additional requests. This knowledge can be obtained by sending a message from a server back to the job dispatcher whenever a job leaves the server previously operating at its maximum bandwidth capacity. Since such updates occur in the background, when a new request arrives, it can be immediately matched to an available server. Furthermore,

unlike the maximum matching algorithm, the RAS scheme does not cause repacking of existing requests in the system.

B. Large System Asymptotics

We now analyze the dynamics of the system under the RAS matching policy. We assume that the requests of each content c arrive according to a Poisson process with rate λ_c, independent of all other processes and the service time of each request is exponentially distributed with unit mean.[1] We define $\rho := \sum_{c\in C}\lambda_c/n\sum_{i\in\mathcal{I},j\in\mathcal{J}}\alpha_{ij}U_i$ to be the load on the system.

We are interested in an asymptotic scaling regime where the number of servers n goes to infinity keeping the load ρ and the proportions α_{ij}, $i\in\mathcal{I}, j\in\mathcal{J}$ fixed. This is achieved by keeping the same catalog C of contents but scaling the arrival rate of each content linearly with n, i.e., $\lambda_c = n\bar{\lambda}_c$ for all $c\in C$. Hence, this scaling represents a scenario where the system size scales proportionally with the demands of the contents. We define $\bar{\lambda} = \sum_{c\in C}\bar{\lambda}_c$. Note that the normalized arrival rates $\hat{\lambda}_c$ remains the same as before.

Before analyzing the dynamics of the caching system, we first determine the fraction of servers in a particular cache configuration under a given allocation policy in the limiting system. Under a given allocation policy, let $q_{ijK}^{(n)}$ denote the fraction of servers having bandwidth U_i ($i\in\mathcal{I}$) and memory size d_j ($j\in\mathcal{J}$) that store all the contents belonging to the set $K\subseteq C$ in the nth system. We call these servers to be in *configuration* (i,j,K) and denote the set of all possible configurations as $\mathcal{L}$. Define $q_{ijK} := \lim_{n\to\infty} q_{ijK}^{(n)}$ for each $(i,j,K)\in\mathcal{L}$, if the limit exists. Clearly, for the `p2p` and the `unif` allocation policies we have $q_{ijK}^{(n)} = q_{ijK} = p_{jK}$, where p_{jK} is defined in (11) and (12), respectively.

The following lemma establishes that q_{ijK} also exists for the `greedy` policy and further characterizes it. We first denote by q_{ijc} the value of q_{ijK} for $K=\{c\}$ and define $\theta_c := \sum_{i\in\mathcal{I},j\in\mathcal{J}}\alpha_{ij}U_i q_{ijc}$ to be the total (normalized) bandwidth capacity allocated to content c. The lemma then states that for $\rho<1$ the capacity allocated to a content is equal to the arrival rate of the content, and for $\rho>1$, unpopular contents are allocated zero capacity. The proof of the lemma is given in Appendix C.

Lemma 1: Under the `greedy` allocation policy, we have $q_{ijK}=0$ for $|K|\geq 2$. Furthermore, for $\rho\leq 1$ we have $\theta_c = \bar{\lambda}_c$, $\forall c\in C$. For $\rho>1$, we have the following: If the popularities are ordered as $\bar{\lambda}_1>\bar{\lambda}_2>\ldots>\bar{\lambda}_m>0$ and c^* is such that $\frac{\bar{\lambda}}{\rho}\in\left(\sum_{c'=1}^{c^*-1}\bar{\lambda}_{c'}-(c^*-1)\bar{\lambda}_{c^*}, \sum_{c'=1}^{c^*}\bar{\lambda}_{c'}-c^*\bar{\lambda}_{c^*+1}\right]$ then

$$\theta_c = \begin{cases} \bar{\lambda}_c - \frac{1}{c^*}\left[\sum_{c'=1}^{c^*}\bar{\lambda}_{c'} - \frac{\bar{\lambda}}{\rho}\right] & \text{if } 1\leq c\leq c^* \\ 0 & \text{if } c^*+1\leq c\leq m \end{cases} \quad (13)$$

Under the assumptions of this section, the dynamics of the nth system can be captured via a suitably constructed Markov process $x^{(n)} = (x_{i,j,K,r}^{(n)}(t), 0\leq r\leq U_i, (i,j,K)\in\mathcal{L}, t\geq 0)$, where $x_{i,j,K,r}^{(n)}(t)$, for $r\in[0,U_i]$, denotes the fraction of servers in configuration $(i,j,K)\in\mathcal{L}$ serving at least r requests at time $t\geq 0$. Clearly, the state space of this process is given by $\mathcal{W}$, where

$$\mathcal{W} := \{w=(w_{i,j,K,r}, 0\leq r\leq U_i, (i,j,K)\in\mathcal{L}): \\ 1 = w_{i,j,K,0}\geq w_{i,j,K,1}\geq\ldots\geq w_{i,j,K,U_i}\geq 0\}.$$

At any time $t\geq 0$, let $x^{(n)}(t)=w\in\mathcal{W}$. Then, at time t, the total number of servers available to serve a request of content $c\in K\subseteq C$ is $\sum_{K':c\in K'}\sum_{i,j} n\alpha_{ij}q_{ijK'}^{(n)}(1-w_{i,j,K',U_i})$. Among these servers there are $n\alpha_{ij}q_{ijK}^{(n)}(w_{i,j,K,r-1}-w_{i,j,K,r})$ servers in configuration (i,j,K) serving exactly $r-1$ ($r\geq 1$) requests. Hence, according to the RAS policy, the Markov process $x^{(n)}$ jumps from a given state $w\in\mathcal{W}$ to the state $w+e_{i,j,K,r}/n\alpha_{ij}q_{ijK}^{(n)}\in\mathcal{W}$ ($r\geq 1$) with rate

$$\sum_{c\in K} n\bar{\lambda}_c \frac{\alpha_{ij}q_{ijK}^{(n)}(w_{i,j,K,r-1}-w_{i,j,K,r})\mathbb{1}(w_{i,j,K,U_i}<1)}{\sum_{K':c\in K'}\sum_{i,j}\alpha_{ij}q_{ijK'}^{(n)}(1-w_{i,j,K',U_i})},$$

where $e_{i,j,K,r}$ denotes the unit vector with unity in position (i,j,K,r). The transition corresponds to an arrival of a request for content $c\in K$. Similarly, the rate of downward transition from $w\in\mathcal{W}$ to $w-e_{i,j,K,r}/n\alpha_{ij}q_{ijK}^{(n)}$ can be computed to be $rn\alpha_{ij}q_{ijK}^{(n)}(w_{i,j,K,r}-w_{i,j,K,r+1})$.

We are interested in the limiting behavior of the process $x^{(n)}$ as $n\to\infty$. We first notice from its transition rates, that $x^{(n)}$ is a *density dependent jump Markov process* (see [26]–[28] for definition) with a *conditional drift* given by the mapping $h:=(h_{i,j,K,r}, 0\leq r\leq U_i, (i,j,K)\in\mathcal{L})$ on $\mathcal{W}$, defined as

$$\begin{aligned} h_{i,j,K,r}(w) &= \lim_{h\downarrow 0}\frac{\mathbb{E}\left[x_{i,j,K,r}^{(n)}(t+h)-x_{i,j,K,r}^{(n)}(t)|x^{(n)}(t)=w\right]}{nh} \\ &= \sum_{c\in K}\bar{\lambda}_c\frac{(w_{i,j,K,r-1}-w_{i,j,K,r})}{\sum_{K':c\in K'}\sum_{i,j}\alpha_{ij}q_{ijK'}(1-w_{i,j,K',U_i})} \\ &\quad\times\mathbb{1}(w_{i,j,K,U_i}<1) - r(w_{i,j,K,r}-w_{i,j,K,r+1}), \\ &\qquad\qquad \text{for } 1\leq r\leq U_i, \quad (14) \end{aligned}$$

and $h_{i,j,K,r}(w)=0$ if $r=0$. For systems in which the drift h is Lipschitz continuous and satisfies certain regularity conditions, the classical results of Kurtz (Theorem 3.1 of [26]) imply that the process $x^{(n)}$ converges in distribution (and hence in probability) to the unique deterministic process $x=(x(t),t\geq 0)$ satisfying $\dot{x}=h(x)$. The process x is called the *mean field limit* or the *fluid limit* of the system. However, since in our case h has discontinuities (due to the presence of the indicator terms), the classical results of Kurtz cannot be applied. To overcome this, we define a set valued map H on $\mathcal{W}$ as the Cartesian product of the set valued maps $H_{i,j,K,r}$ over all possible (i,j,K,r). For each $(i,j,K)\in\mathcal{L}$ we define $H_{i,j,K,r}(w)=\{0\}$ if $r=0$ and

$$\begin{aligned} &H_{i,j,K,r}(w) \\ &= \left[0, \sum_{c\in K}\frac{\bar{\lambda}_c}{\alpha_{ij}q_{ijK}}\right]\mathbb{1}(w_{i,j,K,U_i}=1) \\ &\quad+\sum_{c\in K}\bar{\lambda}_c\frac{(w_{i,j,K,r-1}-w_{i,j,K,r})}{\sum_{K':c\in K'}\sum_{i,j}\alpha_{ij}q_{ijK'}(1-w_{i,j,K',U_i})} \\ &\quad\times\mathbb{1}(w_{i,j,K,U_i}<1) - r(w_{i,j,K,r}-w_{i,j,K,r+1}), \\ &\qquad\qquad \text{for } 1\leq r\leq U_i. \quad (15) \end{aligned}$$

[1] We later show numerically that our results do not depend on the type of service time distribution.

We note that $H_{i,j,K,r}(w) = \{h_{i,j,K,r}(w)\}$ whenever h is continuous at w. At all other points $H(w)$ is the convex hull of all the limit points of $h_{i,j,K,r}(w)$. In the next theorem, whose proof is given in Appendix D, we show that there exists solutions to the differential inclusion (DI) $\dot{x} \in H(x)$ (for a review of differential inclusions see Appendix F)and the process $x^{(n)}$ indeed converges to one of these solutions as $n \to \infty$.

Theorem 3: For any $x_0 \in \mathcal{W}$, the set $\mathcal{S}_{x_0}$ of solutions to the DI $\dot{x} \in H(x)$ with $x(0) = x_0$ is non-empty. Furthermore, if $x^{(n)}(0) \xrightarrow{p} x_0 \in \mathcal{X}$ as $n \to \infty$, then for all $T > 0$ we have

$$\inf_{x \in \mathcal{S}_{x_0}} \sup_{t \in [0,T]} \|x^{(n)}(t) - x(t)\| \xrightarrow{p} 0$$

as $n \to \infty$, where $\xrightarrow{p}$ denotes convergence in probability.

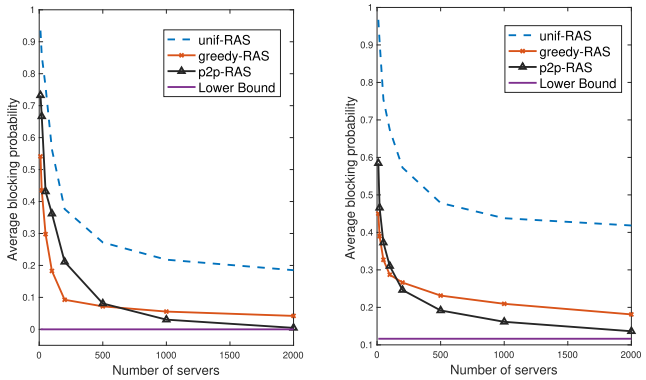
Thus far we have seen that the limiting dynamics of the system for any finite time $t \geq 0$ is described by a solution of the DI $\dot{x} \in H(x)$. We are also interested in the stationary behavior of the limiting system, i.e., the behavior of $x^{(n)}(t)$ as both $n \to \infty$ and $t \to \infty$.[2] Specifically, we are interested in the total number of jobs processed by the system in the stationary regime. The total number of jobs in the nth system at time t is given by $Y^{(n)}(t) = n \sum_{i,j,K} \alpha_{ij} q_{ijK}^{(n)} \sum_{r=1}^{U_i} x_{i,j,K,r}^{(n)}(t)$. Let let $Y^{(n)}(\infty)$ denote its random stationary value of $Y^{(n)}(t)$. Define $y^{(n)}(t) := Y^{(n)}(t)/n$ for all $t \in [0, \infty]$ and $y(t) := \sum_{i,j,K} \alpha_{ij} q_{ijK} \sum_{r=1}^{U_i} x_{i,j,K,r}(t)$. In the next theorem, whose proof is provided in Appendix E, we characterize the limit of $y^{(n)}(\infty)$ as $n \to \infty$ for the `p2p` and the `greedy` algorithms.

Theorem 4: Under the `greedy` allocation policy combined with the RAS matching policy or under the `p2p` allocation policy combined with the RAS matching policy, we have $y(\infty) = \lim_{t\to\infty} y(t) = \min(\bar{\lambda}, \bar{\lambda}/\rho)$. Furthermore, the sequence $(y^{(n)}(\infty))_n$ is tight and $y^{(n)}(\infty) \xrightarrow{p} y(\infty)$.

C. Asymptotic Optimality

Theorem 4 implies that the `p2p` and `greedy` allocation schemes when combined with the RAS matching policy are optimal in the limiting system in terms of maximizing the fraction of accepted requests or the minimizing the fraction of rejected requests. To see this, we find an upper bound on $Y^{(n)}(t)$ for any combination of allocation scheme and matching scheme. A trivial upper bound on $Y^{(n)}(t)$ is clearly the total bandwidth capacity of the system, i.e., $Y^{(n)}(t) \leq n \sum_{ij} \alpha_{ij} U_i = n\bar{\lambda}/\rho$ for all n and $t \in [0, \infty]$. Another upper bound on $Y^{(n)}(\infty)$ can be obtained by comparing the system with another hypothetical caching system in which each server has infinite bandwidth and each content is stored in at least one server. Clearly, this system behaves as an $M/M/\infty$ system that can accept all incoming requests. Hence, the stationary number of requests $\bar{Y}(\infty)$ in this hypothetical system is a Poisson random variable with mean $n\bar{\lambda}$. By a simple coupling argument, it follows that $Y^{(n)}(\infty) \leq \bar{Y}(\infty)$ almost surely for all n. Thus, combining both upper bounds we have $y^{(n)}(\infty) \leq \min\left(\bar{Y}(\infty)/n, \bar{\lambda}/\rho\right)$. Hence, $\limsup_{n\to\infty} y^{(n)}(\infty) \leq \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$. But Theorem 4 shows that for the proposed schemes $\lim_{n\to\infty} y^{(n)}(\infty) = \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$. Hence, the proposed schemes are asymptotically optimal. It is easy to see that the corresponding optimal (minimal) blocking probability is given by $(1 - \min(1, 1/\rho)) = (1 - 1/\rho)^+$.

[2]For finite n, the system always reaches stationarity because the process $x^{(n)}$ is irreducible on a finite state space.



(a) $\rho = 0.8, \eta = 0.8$ (b) $\rho = 1.2, \eta = 1$

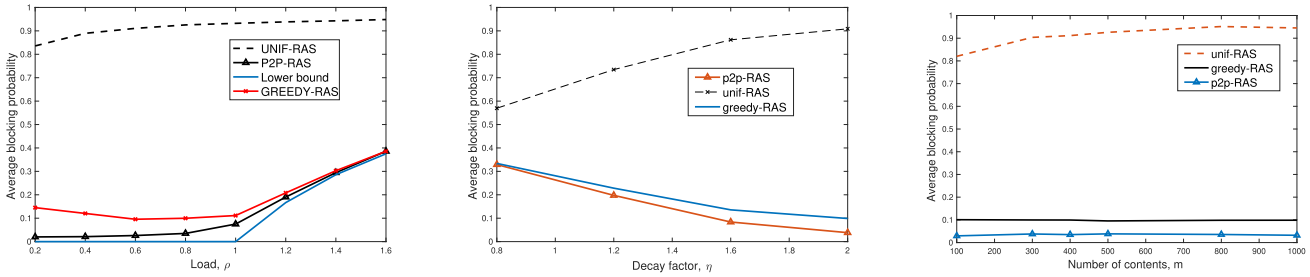
Fig. 1. Average blocking probability as a function of n.

VI. Numerical Results

In this section, we numerically evaluate the performance of the caching system under various allocation policies in conjunction with the RAS matching policy. We consider a system with parameters $d_s = 4, U_s = 10 \forall s \in S$, $m = 500$. The popularities of the contents are chosen according to a Zipf distribution, where the normalized arrival rate $\hat{\lambda}_c$ of any content $c \in C = \{1, 2, \ldots, m\}$ is set to $\hat{\lambda}_c = c^{-\eta} / \sum_{c' \in C} (c')^{-\eta}$. Empirical studies have shown that for video contents, the typical values of η lie in the range from 0.8 to 2 [16], [29]. The system is simulated for different values of n and ρ for 200000 arrivals. In Figures 1(a) and 1(b), we plot the stationary blocking probability of requests as a function the number of servers for different allocation policies combined with the RAS matching policy for $\rho = 0.8$ and $\rho = 1.2$, respectively. The value of η is set to 0.8 and 1 for Figure 1(b) and Figure 1(b), respectively.

We observe that (as predicted by Theorem 4) under both `greedy-RAS` and `p2p-RAS` combinations the blocking probability approaches the optimal lower bound $(1 - 1/\rho)^+$ as n increases. We further observe that for smaller values of n, the `greedy-RAS` combination outperforms the `p2p-RAS` combination while for larger values of n the latter outperforms the former. A possible explanation for this is as follows: for smaller values of n, the `greedy` scheme (similar to the `p2p` scheme) completely fills up the caches of most servers and (unlike the `p2p` scheme) segregates popular contents from unpopular contents by storing them in different servers. As a result the service of the popular contents are not affected by the service of the unpopular contents. However, for larger values of n, under the `greedy` policy most servers store less contents than its memory capacity (as shown by Lemma 1) which results in performance degradation compared to the `p2p` policy (although both schemes are asymptotically optimal).

In Table I, we show the difference between the blocking probability of the finite system $P_{\text{blocking}}^{(n)}$ and the optimal lower



(a) Load (b) Popularity decay factor (c) Number of contents

Fig. 2. Average blocking probability as a function of (a) the load ρ, (b) the popularity decay factor η, and (c) the number of contents m, for the various allocation schemes.

TABLE I

CONVERGENCE OF $P^{(n)}_{\text{BLOCKING}}$

n	$\lvert P^{(n)}_{\text{blocking}} - P_{\text{opt}}\rvert$
10	0.5413
20	0.4348
50	0.2979
200	0.0926
1000	0.0556
2000	0.0421

TABLE II

SENSITIVITY TO THE TYPE OF SERVICE TIME DISTRIBUTIONS

(n, ρ)	Exponential	Constant	Lognormal	Pareto
(400, 0.4)	0.1200	0.1219	0.1241	0.1207
(400, 0.8)	0.0989	0.1028	0.1006	0.0999
(400, 1.2)	0.2084	0.2084	0.2110	0.2092
(400, 1.6)	0.3863	0.3854	0.3878	0.3874
(1000, 0.4)	0.0916	0.0912	0.0915	0.0919
(2000, 0.4)	0.0707	0.0706	0.0690	0.0704

bound $P_{\text{opt}} = (1 - 1/\rho)^+$ as a function of the system size n for the `greedy` algorithm for $\rho = 0.8$, $\eta = 0.8$. We observe that the distance decreases as $O(n^{-1/2})$. The same is observed for the `p2p` policy. This rate of convergence is in accordance with the recent results in the literature on mean field convergence [30].

The blocking probability as a function of the load ρ is shown in Figure 2(a) for fixed $n = 400$, $\eta = 2$. We observe that for the given parameter setting the blocking probability under the `p2p-RAS` combination increases with ρ. However, for the `greedy-RAS` combination the blocking probability first decreases and then increases with the increase in ρ. The reason behind this observation is explained later in this section. In Figure 2(b), we plot the average blocking probability of requests as a function of the decay factor η of the popularity distribution for $\rho = 0.8$ and $n = 400$. We observe that as η increases and the popularity distribution becomes more skewed towards more popular contents, the performance of the `unif` policy degrades as it still treats all contents to be equally popular. On the other hand, for both `greedy` and `p2p` policies the performance improves since under these policies contents with higher popularities are given priority over those with lower popularities.

Next, we study the sensitivity of the system to the type of distribution of the service times of the requests. To this end, we consider the following service time distributions with unit mean: Exponential, Constant, Lognormal with probability density function (PDF) given by $f(x) = (1/x\sqrt{2\pi})e^{-(\ln x + 0.5)^2/2}$, and Pareto with PDF given by $f(x) = 10(0.9)^{10}/x^{11}$. For each distribution we simulate the system (for 160000 arrivals) with $\eta = 2$. The blocking probability of requests is tabulated as a function of n and ρ in Table II for different distributions. We observe that for the same values of n and ρ the blocking probabilities are nearly the same for all distributions. This observation suggests that the system approaches near *insensitivity* for large system sizes. From [31], [32], asymptotic insensitivity is known to hold for similar systems.

In Figure 2(a) and Table II we observe a slight decrease in the blocking probability of the requests with the increase in ρ for the `greedy-RAS` combination in the range $\rho \in [0.4, 0.8]$. While the performance of the RAS matching scheme degrades with the increase in ρ the performance of the `greedy` allocation scheme actually improves. The net effect is a small decrement in the blocking probability. Indeed, as shown in Lemma 1, the fraction of servers containing a content c under the `greedy` scheme is roughly proportional to the arrival rate $\bar{\lambda}_c$ of the content for large systems with identical servers. Since increasing ρ increases $\bar{\lambda}_c$, the fraction of servers containing a particular content also increases. In particular, it increases the availability of highly popular contents which causes the blocking probability to decrease. This phenomenon is absent for the `p2p` scheme since under this scheme the fraction of servers containing a specific content remains the same regardless of the value of ρ.

In Figure 2(c) we plot the average blocking probability of the requests as a function of the number of contents for $\rho = 0.8$, $\eta = 2$, and $N = 400$. We observe that for the `p2p` and the `greedy` policies the average blocking probability remains almost constant with the variation of the number of contents. This is because even though the total number of contents is increasing, only a small fraction of them are highly popular. As a result the addition of more contents does not affect the performance of the system. This also justifies why we scale the arrival rates of the contents instead of the number of contents in Section V.

VII. CONCLUSIONS

We considered the joint problem of content placement and request matching in a distributed network of

content servers. We formulated the problem in an optimization framework and showed that it is NP-hard. We then presented a polynomial-time greedy approximation algorithm for content placement, which is shown to achieve a constant factor approximation guarantee. We then considered the dynamics of the system in the large-systems scaling regime, where we showed that our proposed online matching policy is asymptotically optimal. We employed a new approach based on the theory of differential inclusions to prove our asymptotic results. Many interesting avenues of future work exist. One such challenge is to find the combination of optimal allocation and matching algorithms for systems where both the number of servers and the number of contents scale proportionally to each other.

APPENDIX A
PROOF OF THEOREM 1

We prove the theorem by reducing the 3-partition problem to a decision version of problem (10). The 3-partition problem is defined as follows: Given a finite set G of $3n$ elements, a number $L > 0$ and a mapping $\mathsf{size} : G \to (0, \infty)$ satisfying $\sum_{g\in G} \mathsf{size}(g) = nL$, does their exist n disjoint subsets $G_1, G_2, \ldots, G_n$ of G, each containing three elements, such that for all $1 \le k \le n$, $\sum_{g\in G_k} \mathsf{size}(g) = L$?

We map each element $g \in G$ to a unique content $c \in C$ with $\lambda_c = \mathsf{size}(g)$. Thus we have $m = 3n$ contents and $\sum_c \lambda_c = nL$. The sets $G_1, G_2, \ldots, G_n$ correspond to n servers each of which can store $d_s = 3$ contents and simultaneously serve $U_s = L$ requests. Clearly, for the above defined instance of problem (10), the optimal objective function value is bounded above by nL. We now show that the objective function value exactly equals the upper bound nL if and only if there exists a solution of the 3-partition problem.

Suppose that the optimal objective function value of the above defined instance of (10) is nL and it is achieved at $Z^* = (z^*_{sc})_{s\in S, c\in C}$, i.e., $nL = \sum_{s\in S c\in C} z^*_{sc}$. Since for each $s \in S$, $\sum_{c\in C} z^*_{sc} \le L$, and for each $c \in C$, $\sum_{s\in S} z^*_{sc} \le \lambda_c$, we must have $\sum_{c\in C} z^*_{sc} = L$, $\forall s \in S$ and $\sum_{s\in S} z^*_{sc} = \lambda_c > 0$, $\forall c \in C$ Hence, for each content $c \in C$ there must be one server $s \in S$ such that $z^*_{sc} > 0$, i.e., $\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) \ge 1$. The above implies $\sum_{c\in C} \sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) \ge 3n$. But since every server can store at most three contents, we also have $\sum_{c\in C} \sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) \le 3n$. Hence, we have $\sum_{c\in C} \sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) = 3n$, which implies that $\sum_{s\in S} \mathbb{1}(z^*_{sc} > 0) = 1$, $\forall c \in C$ and $\sum_{c\in C} \mathbb{1}(z^*_{sc} > 0) = 3$, $\forall s \in S$. Hence, a solution of the 3-partition problem is found.

Conversely, suppose a solution to the 3-partition problem exists. Using this solution we now construct an optimal solution of problem (10). Denote the n disjoint subsets of G as $G_1, G_2, \ldots, G_n$ and associate a distinct server s_k with each set G_k, for $k = 1, 2, \ldots, n$. Also associate a distinct content c with each element $g \in G$. We store the content c in server s_k if and only if the element g belongs to the partition G_k and in this case we set $z_{s_k c} = \mathsf{size}(g)$. Note that the memory constraints at each server is satisfied since each server stores three distinct contents. Now since the element g is in only one of the G_i's, we also have $\sum_{i=1}^n z_{s_i c} = \mathsf{size}(g) = \lambda_c$. Furthermore, we have $\sum_{c\in C} z_{s_k c} = \sum_{g\in G_i} \mathsf{size}(g) = L$ for all $k = 1, 2, \ldots, n$. Hence, $\sum_{k=1}^n \sum_{c\in C} z_{s_k c} = nL$, which is the optimal value of the objective function of the instant of problem (10) under consideration. □

APPENDIX B
PROOF OF THEOREM 2

To prove the theorem we first introduce a slightly modified version of problem (10) by adding a constraint which requires that only pairs (s, c) belonging to a given set $\Gamma \subseteq S \times C$ can be assigned a non-zero value of z_{sc}, i.e., $z_{sc} = 0$ for all $(s, c) \notin \Gamma$. This modified problem is denoted as $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma)$ and its optimal value is denoted as $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma)$. We note that $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$ is a special case of $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma)$ with $\Gamma = S \times C$. Clearly, if $\mathcal{U} \le \mathcal{U}'$, $\Lambda \le \Lambda'$, $\mathcal{D} \le \mathcal{D}'$, $\Gamma \subseteq \Gamma'$, then $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D}, \Gamma) \le \texttt{JAM}^*(\mathcal{U}', \Lambda', \mathcal{D}', \Gamma')$. Let $H^* = \{(s, c) \in S \times C : z^*_{sc} > 0\}$ denote the collection of (s, c) pairs that are assigned non-zero values of z^*_{sc} in the optimal solution of $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$. Clearly, we have $\texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D}, H^*) = \texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$.

We now generate a sequence of tuples $(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k, G_k)$ for each iteration $k \ge 0$ of the `greedy` algorithm (applied on $\texttt{JAM}(\mathcal{U}, \Lambda, \mathcal{D})$) as follows: For $k = 0$ we set $\mathcal{U}_0 = \mathcal{U}, \Lambda_0 = \Lambda, \mathcal{D}_0 = \mathcal{D}, H^*_0 = H^*$, $G_0 = S \times C$. For subsequent values of k, let $f_k (= \texttt{MatchedFlow})$ denote the flow found by the `greedy` algorithm in its k^{th} iteration and let (s_k, c_k) $(= (s^*, c^*))$ denote the corresponding server-content pair. We set

$$\begin{aligned} G_{k+1} &= G_k - (s_k, c_k), \\ \mathcal{U}_{k+1} &= \mathcal{U}_k - f_k e^{(n)}_{s_k}, \\ \Lambda_{k+1} &= \Lambda_k - f_k e^{(m)}_{c_k}, \\ \mathcal{D}_{k+1} &= \mathcal{D}_k - e^{(n)}_{s_k}, \\ H^*_{k+1} &= H^*_k - (s_k, c_k), \end{aligned}$$

where $e^{(l)}_r$ denotes the l-dimensional standard unit vector having one in the r^{th} component.

Note that it may so happen that $(s_k, c_k) \notin H^*_k$ for some k (but (s_k, c_k) will always be in G_k). In such cases, the above update sets $H^*_{k+1} = H^*_k$. Hence, $H^*_k \subseteq G_k$ is maintained for all k. Therefore, we have $\texttt{JAM}^*(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k) \le \texttt{JAM}^*(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, G_k)$. If the `greedy` algorithm terminates in the q^{th} iteration, then we must have $f_q = 0$ and $\texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, D_q, G_q) = 0$. Therefore, $\texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, \mathcal{D}_q, H^*_q) = 0$. Furthermore, for each iteration k we prove that the following inequality

$$\begin{aligned} &\texttt{JAM}^*(\mathcal{U}_k, \Lambda, \mathcal{D}_k, H^*_k) \\ &\qquad - \texttt{JAM}^*(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H^*_{k+1}) \le 2f_k, \quad (16) \end{aligned}$$

holds. To see the above, first let us consider the case when $(s_k, c_k) \in H^*_k$. Hence, $(s_k, c_k) \notin H^*_{k+1}$. Let $v \ge 0$ be the flow assigned to the (s_k, c_k) pair in the optimal solution of $\texttt{JAM}(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k)$. We have

$$\begin{aligned} &\texttt{JAM}^*(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H^*_k) \\ &\quad = v + \texttt{JAM}^*(\mathcal{U}_k - v e^{(n)}_{s_k}, \Lambda_k - v e^{(m)}_{c_k}, \mathcal{D}_{k+1}, H^*_{k+1}) \\ &\quad \le v + \texttt{JAM}^*(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H^*_{k+1}) \\ &\qquad + 2(f_k - v) \end{aligned}$$

The last inequality holds since no more than $2(f_k - v) \geq 0$ additional flow can be matched under constraints given by the tuple $(\mathcal{U}_k - ve_{s_k}^{(n)}, \Lambda_k - ve_{c_k}^{(m)}, \mathcal{D}_{k+1}, H_{k+1}^*)$ as compared to constrains given by the tuple $(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H_{k+1}^*)$. Inequality (16) hence follows for $(s_k, c_k) \in H_k^*$. Now consider the case when $(s_k, c_k) \notin H_k^*$. Hence, $H_k^* = H_{k+1}^*$. Clearly, no more than $2f_k$ additional flow can be matched under the constraints given by $(\mathcal{U}_k, \Lambda_k, \mathcal{D}_k, H_k^*)$ as compared to the constraints given by $(\mathcal{U}_{k+1}, \Lambda_{k+1}, \mathcal{D}_{k+1}, H_{k+1}^*)$. Hence, (16) holds in this case also. Summing (16) for $k = 0, 1, \ldots, q-1$ we obtain

$$\texttt{JAM}^*(\mathcal{U}_0, \Lambda_0, \mathcal{D}_0, H_0^*) - \texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, \mathcal{D}_q, H_q^*) \leq 2\sum_{k=0}^{q-1} f_k. \tag{17}$$

This completes the proof since $\sum_{k=0}^{q-1} f_k$ is the output of the greedy algorithm, $\texttt{JAM}^*(\mathcal{U}_0, \Lambda_0, \mathcal{D}_0, H_0^*) = \texttt{JAM}^*(\mathcal{U}, \Lambda, \mathcal{D})$, and $\texttt{JAM}^*(\mathcal{U}_q, \Lambda_q, \mathcal{D}_q, H_q^*) = 0$. □

APPENDIX C
PROOF OF LEMMA 1

First we note that for sufficiently large n, we have $n\bar{\lambda}_c > U_i$ for all $c \in C$ and all $i \in \mathcal{I}$. Hence, to allocate more than one content to the cache of a server the greedy algorithm must reach a stage where the remaining flow of each content is less than $U_{\max} := \max_{i\in\mathcal{I}} U_i$. Since at least $U_{\min} := \min_{i\in\mathcal{I}} U_i$ flow can be matched to each server, the maximum number of servers which can be assigned a non-zero flow after this stage is $mU_{\max}/U_{\min}$. These are the only servers which can be allocated more than one contents. Therefore, the fraction of servers assigned more than two contents is at most $mU_{\max}/nU_{\min}$ which approaches zero as $n \to \infty$. This shows that the probability of a server storing more than one content approaches zero as $n \to \infty$, i.e., $q_K^{(ij)} = 0$ for $|K| \geq 2$.

Next, consider the case $\rho \leq 1$. Again, for sufficiently large n, we have $n\bar{\lambda}_c > U_{\max}$ for all $c \in C$. In this case, the greedy algorithm cannot terminate before the remaining flow for each content becomes less than or equal to $U_{\max}$. To prove this, let us assume the converse, i.e., the greedy algorithm terminates when the remaining flows for some contents are still strictly above $U_{\max}$. This implies that the greedy algorithm terminated because the remaining flows of each server has become zero. Clearly, for this to happen we must have $n\bar{\lambda} > n\sum_{i,j} \alpha_{ij} U_i$, i.e., $\rho > 1$, which is a contradiction. Therefore, the greedy algorithm terminates with less than $U_{\max}$ remaining flow for each content. Thus, the fraction of the total flow $n\bar{\lambda}$ which remains unmatched is at most $\frac{mU}{n\bar{\lambda}}$, which approaches to zero as $n \to \infty$. Hence, in the limiting system, the whole flow $n\bar{\lambda}_c$ of each content $c \in C$ is matched. Since $n\theta_c$ denotes the total flow of content c assigned to all the servers combined, we must have $\theta_c = \bar{\lambda}_c$.

For $\rho > 1$, it is easy to see that the greedy algorithm terminates when remaining flows of all the servers become zero. Furthermore, at termination, all the contents, which have been chosen at least once by the algorithm in some iteration, have the same remaining flow. Let this flow be equal to f and let the contents chosen by the algorithm at least once be $c = 1, 2, \ldots, c^*$ for some $c^* \leq m$. Then, we must have $n\bar{\lambda}_{c^*+1} \leq f < n\bar{\lambda}_{c^*}$. Also, since the total matched flow $n\sum_{c'=1}^{c^*} \bar{\lambda}_{c'} - kf$ combining all the contents is equal to the total capacity $\bar{\lambda}/\rho$ of the system, we have $f = \frac{n}{c^*}\left(\sum_{c'=1}^{c^*} \bar{\lambda}_{c'} - \frac{\bar{\lambda}}{\rho}\right)$. The matched flow for each content $c = 1, 2, \ldots, c^*$ is therefore $n\theta_c = n\bar{\lambda}_c - f$ and for each content $c > c^*$ is $\theta_c = 0$. This completes the proof of the lemma. □

APPENDIX D
PROOF OF THEOREM 3

We recall from Theorem 1 of [33] that the DI $\dot{x} \in H(x)$ has at least one solution x with $x(0) = x_0 \in \mathcal{W}$ if 1) for each $w \in \mathcal{W}$ the set $H(w)$ is non-empty, closed, convex; 2) $\|H(w)\| := \sup\{\|z\|_2 : z \in H(w)\} < D(1 + \|w\|)$ for some constant $D > 0$; 3) H is upper semi-continuous, i.e., at each $w \in \mathcal{W}$, $w_n \to w, z_n \in H(w_n)$, and $\lim_{n\to\infty} z_n = z$ implies $z \in H(w)$. Furthermore, the density dependent Markov process $x^{(n)}$ with limiting drift h converges in probability to the solution of the DI $\dot{x} \in H(x)$ if $H(w) = \mathsf{conv}\left(\mathsf{acc}_{w_k \to w} h(w_k)\right)$, where $\mathsf{conv}(V)$ denotes the closure of convex hull containing the set V and $(\mathsf{acc}_{w_k\to w} h(w_k))$ denotes the set of accumulation points of the sequence $(h(w_k))$ for $w_k \to w$.

From (15), it follows directly that $H(w)$ is nonempty, closed and convex for each $w \in \mathcal{W}$. Furthermore, for each $(i, j, K) \in \mathcal{L}$ and $r \in \mathbb{Z}_+$ we have $0 \leq \bar{\lambda}_c \frac{(w_{i,j,K,r-1} - w_{i,j,K,r})}{\sum_{K':c\in K'} \sum_{i,j} \alpha_{ij} q_{ijK'} (1 - w_{i,j,K',U_i})} \leq \bar{\lambda}_c/\alpha_{ij} q_{ijK}$. Hence, from (15) we have that for any $z_{i,j,K,r} \in H_{i,j,K,r}(w)$

$$z_{i,j,K,r} \leq D_{i,j,K} := \sum_{c\in K} \frac{\bar{\lambda}_c}{\alpha_{ij} q_{ijK}} + \max_{i\in\mathcal{I}} U_i$$

Therefore, $\|H(w)\| \leq D \leq D(1 + \|w\|_2)$, where $D := \sqrt{\sum_{(i,j,K)\in\mathcal{L}} U_i D_{i,j,K}^2} > 0$. We also note that $H_{i,j,K,r}(w)$ is continuous if $w_{i,j,K,U_i} < 1$ and the compact set $[0, \bar{\lambda}_c/\alpha_{ij} q_{ijK}]$ contains all limit points of $\bar{\lambda}_c \frac{(w_{i,j,K,r-1} - w_{i,j,K,r})}{\sum_{K':c\in K'} \sum_{i,j} \alpha_{ij} q_{ijK'} (1 - w_{i,j,K',U_i})}$ as $w_{i,j,K,U_i} \to 1$. Hence, the set valued mapping H is upper semi-continuous. By definition it follows that $H(w) = \mathsf{conv}\left(\mathsf{acc}_{w_k\to w} h(w_k)\right)$. Therefore, the statement of the theorem follows from Theorem 1 of [33]. □

APPENDIX E
PROOF OF THEOREM 4

We first consider the greedy scheme combined with the RAS scheme. We define

$$y_c(t) := \sum_{K:c\in K} \sum_{i,j} \alpha_{ij} q_{ijK} \sum_{r=1}^{U_i} x_{i,j,K,r}(t),$$

for all $t \geq 0$. For the greedy algorithm, we have from Lemma 1 that $q_{ijK} = 0$ for $|K| > 1$. Hence, we have $y_c(t) = \sum_{i,j} \alpha_{ij} q_{ijc} \sum_{r=1}^{U_i} x_{i,j,K,r}(t)$ and $y(t) = \sum_{c\in C} y_c(t)$. Further, in (15) substituting $K = \{c\}$, multiplying by $\alpha_{ij} q_{ijc}$ and summing over i and j we obtain

$$\dot{y}_c(t) \in H_c(y_c) := [0, \bar{\lambda}_c]\mathbb{1}(y_c = \theta_c) + \bar{\lambda}_c \mathbb{1}(y_c < \theta_c) - y_c. \tag{18}$$

It can be easily verified that $(z_1 - z_2)(w_1 - w_2) \leq 0$ for all $w_1, w_2 \in \mathbb{R}$ and for all $z_1 \in H_c(w_1), z_2 \in H_c(w_2)$. In other words, H_c is one-sided Lipschitz (OSL) with Lipschitz constant $L = 0$. Therefore, the DI (18) has a unique solution. It can also be verified that for any $y(0) = y_0 \in [0, \theta_c]$,

$$y_c(t) = \tilde{y}_c(t)\mathbb{1}\left(\tilde{y}_c(t) < \theta_c\right) + \theta_c \mathbb{1}\left(\tilde{y}_c(t) \geq \theta_c\right),$$

where $\tilde{y}_c(t) = \bar{\lambda}_c + (y_0 - \bar{\lambda}_c)e^{-t}$, is a solution of the DI (18). Hence, it must be the only solution. Since from Lemma 1 we have for $\rho \leq 1$, $\theta_c = \bar{\lambda}_c$ for all $c \in C$, it follows that $y_c(\infty) = \lim_{t\to\infty} y_c(t) = \bar{\lambda}_c$, or $y(\infty) = \bar{\lambda}$. For $\rho > 1$ again from Lemma 1 we have that $\theta_c < \bar{\lambda}_c$ for all $c = 1 : c^*$ and $\theta_c = 0$ for $c = c^* + 1 : m$. Hence, $y_c(\infty) = \theta_c$ for $c = 1 : c^*$ and $y_c(\infty) = 0$ for $c = c^* + 1 : m$. Thus, $y(\infty) = \sum_{c\in C} y_c(\infty) = \sum_{c=1}^{c^*} \theta_c = \bar{\lambda}/\rho$.

Now, we consider the p2p scheme combined with the RAS scheme. It is sufficient to consider the case with $d_s = 1$ for all $s \in S$. This is because the performance of the caching system can only improve with the increase in the memory sizes. For this case we have $q_{ijc} = \hat{\lambda}_c$ for all $c \in C$ and $q_{ijK} = 0$ if $|K| > 1$. Hence, as before we have $y(t) = \sum_{c\in C} y_c(t)$ and $y_c(t) = \tilde{y}_c(t)\mathbb{1}\left(\tilde{y}_c(t) < \theta_c\right) + \theta_c \mathbb{1}\left(\tilde{y}_c(t) \geq \theta_c\right)$. In this case, $\theta_c = \bar{\lambda}_c/\rho$. We thus have, for $\rho \leq 1$, $\bar{\lambda}_c \leq \theta_c$ and for $\rho > 1$ $\theta_c < \lambda_c$. Hence, $y_c(\infty) = \lim_{t\to\infty} y_c(t) = \lambda_c$ for $\rho \leq 1$ and $y_c(\infty) = \theta_c = \lambda_c/\rho$ for $\rho > 1$. Thus, $y(\infty) = \min\left(\bar{\lambda}, \bar{\lambda}/\rho\right)$.

Since $y^{(n)}(\infty) \leq \bar{\lambda}/\rho$ uniformly for all n, it follows that the sequence $(y^{(n)}(\infty))_n$ is tight. From Theorem 3 and the uniqueness of solution of $y(t)$ we have that $y^{(n)} \to y$. Hence, every limit point of the sequence of stationary measures of $y^{(n)}$ must be an invariant measure of the process y. Since $y(\infty)$ is the unique, globally asymptotically stable stationary point of the process y it follows that the only invariant measure for the process y is the Dirac measure concentrated at $y(\infty)$. Thus, all limit points of the sequence of stationary measures of $y^{(n)}$ must coincide with the Dirac measure at $y(\infty)$, i.e., $\lim_{n\to\infty} y^{(n)}(\infty) = y(\infty)$. □

APPENDIX F
REVIEW OF DIFFERENTIAL INCLUSIONS

We recall from [34], [35] that an absolutely continuous process $y(\cdot)$ in $\mathbb{R}^d$ defined on $I \subset R$ is called a solution of the differential inclusion (DI) $\dot{y} \in F(y)$ with initial condition $y(0) = y_0 \in \mathbb{R}^d$, if for the set valued mapping $F(\cdot) : \mathbb{R}^d \to 2^{\mathbb{R}^d}$ there exists a mapping $\varphi : I \to \mathbb{R}^d$ such that for all $t \in I$

$$y(t) = y_0 + \int_0^t \varphi(s)ds \tag{19}$$

and $\varphi(s) \in F(y(s))$ for almost every $s \in I$. We say that the DI corresponds to drift function $f : \mathbb{R}^d \to \mathbb{R}^d$ if for each $y \in \mathbb{R}^d$ we have $F(y) = \mathsf{conv}\left(\mathsf{acc}_{y_k\to y} f(y_k)\right) \subset \mathbb{R}^d$, where $\mathsf{conv}(A)$ denotes the closure of convex hull containing the set A and $(\mathsf{acc}_{y_k\to y} f(y_k))$ denotes the set of accumulation points of the sequence $(f(y_k))$ for $y_k \to y$. Clearly, if f is continuous at point $y \in \mathbb{R}^d$, then according to the above definition $F(y) = \{f(y)\}$ and at points of y of discontinuities $F(y)$ is the convex hull of all the limit points of the sequence $(f(y_k))_k$, where $y_k \to y$ as $k \to \infty$.

The DI $\dot{y} \in F(y)$ has solutions in $\mathbb{R}^d$ on the interval $[0, \infty)$ if 1) for each $y \in \mathbb{R}^d$, the set $F(y) \subset \mathbb{R}^d$ is closed, convex, non-empty; 2) $\|F(y)\| := \sup\{\|z\|_2 : z \in F(y)\} < K(1 + \|y\|)$ for some constant $K > 0$; 3) F is upper semicontinuous, i.e., at each $y \in R^d$, $y_k \to y$, $z_k \in F(y_k)$, and $\lim_{k\to\infty} z_k = z$ implies $z \in F(y)$. Furthermore, if F is one-sided Lipschitz, i.e., satisfies for all $y, y' \in \mathbb{R}^d$ and for all $z \in F(y), z' \in F(y')$

$$\langle y - y', z - z' \rangle \leq L\|y - y'\|^2 \tag{20}$$

for some constant $L \in \mathbb{R}$, then the solution to $\dot{y} \in F(y)$ is unique.

REFERENCES

[1] A. Mukhopadhyay, N. Hegde, and M. Lelarge, "Optimal content replication and request matching in large caching systems," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 288–296.

[2] "Cisco visual networking index: Forecast and methodology 2016–2021," Cisco, San Jose, CA, USA, White Paper, 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/global-cloud-index-gci/white-paper-c11-738085.pdf

[3] J. Dilley *et al.*, "Globally distributed content delivery," *IEEE Internet Comput.*, vol. 6, no. 5, pp. 50–58, Sep./Oct. 2002.

[4] F. P. Kelly, "Loss networks," *Ann. Appl. Probab.*, vol. 1, no. 3, pp. 319–378, 1991.

[5] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang, "Vivisecting youtube: An active measurement study," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2521–2525.

[6] M. A. Salahuddin, J. Sahoo, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on content placement algorithms for cloud-based content delivery networks," *IEEE Access*, vol. 6, pp. 91–114, 2017.

[7] B. Tan and L. Massoulié, "Optimal content placement for peer-to-peer video-on-demand systems," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 566–579, Apr. 2013.

[8] M. Leconte, M. Lelarge, and L. Massoulié, "Bipartite graph structures for efficient balancing of heterogeneous loads," *ACM SIGMETRICS Perform. Eval. Rev. Perform. Eval. Rev.*, vol. 40, no. 1, pp. 41–52, Jun. 2012.

[9] S. Tewari and L. Kleinrock, "Proportional replication in peer-to-peer networks," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–2.

[10] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[11] M. Dehghan *et al.*, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 936–944.

[12] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Proactive retention-aware caching with multi-path routing for wireless edge networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1286–1299, Jun. 2018.

[13] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Multicast-aware caching for small cell networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2014, pp. 2300–2305.

[14] A. Khreishah, J. Chakareski, and A. Gharaibeh, "Joint caching, routing, and channel assignment for collaborative small-cell cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2275–2284, Aug. 2016.

[15] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, Jun. 2018.

[16] S. Moharir and N. Karamchandani, "Content replication in large distributed caches," 2016, *arXiv: 1603.09153*. [Online]. Available: https://arxiv.org/abs/1603.09153

[17] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proc. 22nd Annu. ACM Symp. Theory Comput.*, New York, NY, USA, 1990, pp. 352–358.

[18] A. Mastin and P. Jaillet, "Greedy online bipartite matching on random graphs," 2013, *arXiv:1307.2536*. [Online]. Available: https://arxiv.org/abs/1307.2536

[19] A. L. Stolyar, "Large-scale heterogeneous service systems with general packing constraints," *Adv. Appl. Probab.*, vol. 49, no. 1, pp. 61–83, 2017.

[20] S. Moharir, J. Ghaderi, S. Sanghavi, and S. Shakkottai, "Serving content with unknown demand: The high-dimensional regime," *ACM SIGMETRICS Perform. Eval. Rev. Perform. Eval. Rev.*, vol. 42, no. 1, pp. 435–447, Jun. 2014.
[21] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM J. Optim.*, vol. 12, no. 2, pp. 479–502, Feb. 2002.
[22] W. Cook and W. R. Pulleyblank, "Linear systems for constrained matching problems," *Math. Oper. Res.*, vol. 12, no. 1, pp. 97–120, 1987.
[23] E. Anshelevich, O. Bhardwaj, and M. Usher, "Friend of my friend: Network formation with two-hop benefit," *Theory Comput. Syst.*, vol. 57, no. 3, pp. 711–752, Oct. 2015.
[24] B. Korte and D. Hausmann, "An analysis of the greedy heuristic for independence systems," in *Algorithmic Aspects Combinatorics*, vol. 2, B. Alspach, P. Hell, and D. J. Miller, Eds. 1978, pp. 65–74.
[25] D. Hausmann, B. Korte, and T. A. Jenkyns, *Worst Case Analysis Of Greedy Type Algorithms For Independence Systems*. Berlin, Germany: Springer, 1980, pp. 120–131.
[26] T. G. Kurtz, "Solutions of ordinary differential equations as limits of pure jump Markov processes," *J. Appl. Probab.*, vol. 7, no. 1, pp. 49–58, 1970.
[27] M. Mitzenmacher, "The power of two choices in randomized load balancing," Ph.D. dissertation, Univ. California Berkeley, Berkeley, CA, USA, 1996.
[28] A. Mukhopadhyay, A. Karthik, and R. R. Mazumdar, "Randomized assignment of jobs to servers in heterogeneous clusters of shared servers for low delay," *Stochastic Syst.*, vol. 6, no. 1, pp. 1–250, 2016.
[29] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 126–134.
[30] L. Ying, "On the approximation error of mean-field models," *SIGMETRICS Perform. Eval. Rev. Perform. Eval. Rev.*, vol. 44, no. 1, pp. 285–297, Jun. 2016.
[31] A. Mukhopadhyay, A. Karthik, R. R. Mazumdar, and F. M. Guillemin, "Mean field and propagation of chaos in multi-class heterogeneous loss models," *Perform. Eval.*, vol. 91, pp. 117–131, Sep. 2015.
[32] T. Vasantam, A. Mukhopadhyay, and R. R. Mazumdar, "Insensitivity of the mean-field limit of loss systems under power-of-d routing," Aug. 2017, *arXiv:1708.09328*. [Online]. Available: https://arxiv.org/abs/1708.09328
[33] N. Gast and B. Gaujal, "Mean field limit of non-smooth systems and differential inclusions," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 2, pp. 30–32, Oct. 2010.
[34] M. Kunze, *Non-Smooth Dynamical Systems*. Berlin, Germany: Springer, 2000.
[35] N. Gast and B. Gaujal, "Markov chains with discontinuous drifts have differential inclusion limits," *Perform. Eval.*, vol. 69, no. 12, pp. 623–642, 2012.

Arpan Mukhopadhyay received the B.E. degree in electronics and telecommunication engineering from Jadavpur University, Calcutta, India, in 2009, the M.E. degree in telecommunications from the Indian Institute of Science, Bengaluru, India, in 2011, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2016.

He is currently an Assistant Professor with the Department of Computer Science, University of Warwick, U.K. Before joining the University of Warwick, he spent two years with EPFL, Switzerland, and one year with INRIA, Paris, as a Post-Doctoral Researcher. His research interests include applied probability, stochastic processes, algorithm design, and optimization with applications to cloud computing systems, caching systems, wireless networks, social networks, and smart grids. He was a recipient of Best Paper Awards from the IFIP Performance 2015 Conference and the International Teletraffic Congress (ITC) 2015, and he received the Rising Scholar Award at the International Teletraffic Congress 2018 for his contributions to mean field analysis of large heterogeneous networks.

Nidhi Hegde is currently a Research Team Lead with Borealis AI, Canada. Her team is currently focused on developing machine learning models that include privacy and ethics. She has previously been at Bell Labs, Technicolor Labs, and France Telecom, where she worked on algorithms for resource allocation.

Marc Lelarge graduated from École Polytechnique, Palaiseau, France, and received the Engineering degree from the École Nationale Supérieure des Télécommunications, Paris, France, and the Ph.D. degree in applied mathematics from École Polytechnique in 2005. He is a Researcher with INRIA leading the DYOGENE Research Team, which is part of the Computer Science Department, École Normale Supérieure des Télécommunications. He is also a Lecturer in deep learning with École Polytechnique and the École Normale Supérieure des Télécommunications. His research interests include machine learning, deep learning, graphs, and data analytics.

Dr. Lelarge received the NetGCoop 2011 Best Paper Award with his Ph.D. student E. Coupechoux, the 2012 SIGMETRICS Rising Star Researcher Award, and the 2015 Best Publication in Applied Probability Award with M. Bayati and A. Montanari for their work on compressed sensing.